TRANSPORTATION ANALYSIS SIMULATION SYSTEM (TRANSIMS)

Version: TRANSIMS-LANL-1.0

VOLUME 6 – INSTALLATION

28 May 1999

LA-UR 99-2580

TRANSIMS

Version: TRANSIMS-LANL-1.0

VOLUME 6 – INSTALLATION

28 May 1999

LA-UR-99-2580

The following persons contributed to this document:

C. L. Barrett*
R. J. Beckman*
K. P. Berkbigler*
K. R. Bisset*
B. W. Bush*
S. Eubank*
J. M. Hurford*
G. Konjevod*
D. A. Kubicek*
M. V. Marathe*
J. D. Morgeson*
M. Rickert*
P. R. Romero*
L. L. Smith*
M. P. Speckman**
P. L. Speckman**
P. E. Stretz*
G. L. Thayer*
M. D. Williams*

* Los Alamos National Laboratory, Los Alamos, NM 87545
** National Institute of Statistical Sciences, Research Triangle Park, NC

Acknowledgments

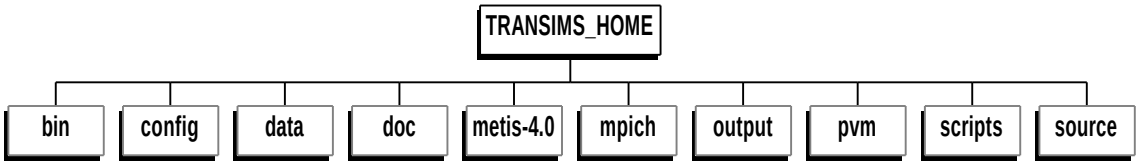
This work was supported by the U. S. Department of Transportation (Assistant Secretary for Transportation Policy, Federal Highway Administration, Federal Transit Administration), the U. S. Environmental Protection Agency, and the U. S. Department of Energy as part of the Travel Model Improvement Program.

CONTENTS

1. WHAT'S ON THE CD-ROM?

The TRANSIMS distribution on the CD-ROM contains the directory structure shown below. The root directory of the distribution, after it is installed, will be referred to as *TRANSIMS_HOME*.

TRANSIMS Directory Structure



1.1 Distribution Description

1.1.1 bin Directory

The ***bin*** directory contains the executable programs compiled for Linux/Intel:

- *TRANSIMS_HOME/buildit* – C Shell script files used to compile the TRANSIMS components

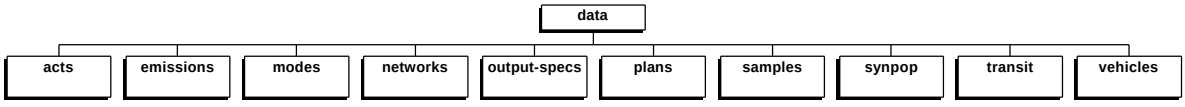
1.1.2 config Directory

The ***config*** directory contains the TRANSIMS configuration files for running TRANSIMS programs on the networks supplied with this release. Configuration files define parameters for all TRANSIMS programs. The format of the files is `<key value>`. Refer to sections in Volume 3—*Files* for explanations of configuration file keys.

1.1.3 data Directory

The ***data*** directory contains subdirectories of the data needed to run the TRANSIMS components.

Data Directory



- ***acts*** – ample activity sets.
- ***emissions*** – data files used by the Emissions Estimator
- ***modes*** – TRANSIMS mode files.
  - *calnet.modes* - mode file for Multimode Network
  - *local.modes* - mode file for Local Streets Network

  - *output_test.modes* - mode file for output test network

- ***networks*** – network tables for the TRANSIMS networks. The following networks are supplied with this release.
  - *Multi-mode* - multimode feedback test network using simplified population and activity generation (*calnet_ network table prefix*)
  - *local streets* - test network of TRANSIMS components (*Test_Local* network table prefix)
  - *Output Test* - network to test TRANSIMS microsimulation output specifications (*Test_ network* table prefix)
  - *Microsimulation Calibration Networks* - one- and three-lane freeways, two-lane left turn, two-lane merge, and tee intersection networks. (*Calibration_1, Calibration_2* network table prefixes)
- ***output-specs*** – TRANSIMS output specification files for output collection on nodes and links for the TRANSIMS networks.
- ***plans*** – route plans for travelers on the TRANSIMS networks.
  - *microsimulation calibration plan sets* (*Freeway1.plans, Freeway3.plans, Left2.plans, Merge2.plans, Tee.plans)*
  - *output test plan set* (*OutputTest.plans)*
  - *sample plan sets on calnet and local streets networks* (*calnet.plans, local.plans)*
- ***samples*** – sample output data from the TRANSIMS components.
- ***synpop*** – data files used by the Population Synthesizer. Includes the PUMS (*pums*), STF3A (*stf*), and MABLE/GEOCORR (*NewMexico_MABLE.csv*) data used to generate a block group to be located on the TRANSIMS local streets network.
- ***transit*** – transit (bus) route and schedule data for the Multimode and *output_test* networks.
- ***vehicles*** –TRANSIMS vehicle files for microsimulation calibration networks (*Freeway1.vehicles, Freeway3.vehicles, Left2.vehicles, Merge2.vehicles,* and *Tee.vehicles*) and the output test network (*OutputTest.vehicles*). Sample vehicle file for the local streets network (*local.vehicles*) and Multimode Network (*Calnet.vehicles*).

1.1.4 doc Directory

The ***doc*** directory contains the TRANSIMS documents.

- Volume 0—*Overview*
- Volume 2—*Software*
- Volume 3—*Files*
- Volume 4—*Case Studies*

- Volume 6—*Installation*

1.1.5 metis-4.0 Directory

The ***metis-4.0*** directory contains the distribution of *metis-4.0* used by the Traffic Microsimulator to partition the transportation network among the available CPUs.

1.1.6 mpich Directory

The ***mpich*** directory contains Message Passing Interface (MPI) distribution 1.1.1, used by the Traffic Microsimulator for parallel communications.

1.1.7 output Directory

The ***output*** directory is the location where Traffic Microsimulator output will be placed by default.

1.1.8 pvm Directory

The ***pvm*** directory contains the distribution of PVM 3.4.0 used by the Traffic Microsimulator.

1.1.9 scripts Directory

The ***scripts*** directory contains the Perl and C-Shell scripts used to run the TRANSIMS components.

- Perl scripts used to run the Traffic Microsimulator (*CommonInterface.pl, Communcation.pl, ExpandPath.pl, MPIUtil.pl, Msim.pl, PVMUtil.pl, TransimsUtil.pl, pvmd.pl*)
- C shell script used to run feedback experiments on the Multimode Network (*expt.sh, GenCalnetBuses.sh*)

1.1.10 source Directory

The ***source*** directory contains the following source code subdirectories for the TRANSIMS components.

- *ACT* – TRANSIMS Activity Generator
- *ACTL* – Simplified Activity Generator
- *CA, PAR, COMMON, TBX* – Traffic Microsimulator
- *CALIB* – Traffic Microsimulator calibration output filters
- *ENV* – Emissions Estimator
- *GBL* – Global definitions and methods used by other TRANSIMS components
- *IO* – TRANSIMS file interface definitions and methods
- *ITDB* – Iteration Database

- *NET* – TRANSIMS network
- *OUT* – TRANSIMS Output
- *PLAN* – TRANSIMS route plan definition and methods
- *POPL* - Simplified Population Generator
- *ROUTER, THREADS* –Route Planner
- *SEL* – TRANSIMS Selector
- *SYNPOP* – Population Synthesizer
- *TRANSIT* – Transit methods
- *TRAV* – Traveler definitions
- *VEH* – TRANSIMS Vehicle file generator
- *VIS, mui* – Output Visualization

The ***source*** directory also contains the following files for the TRANSIMS components.

- *Makefile* – symbolic link to *source/Makefile.main*. Used for compilation of TRANSIMS components in the *TRANSIMS_HOME* directory.
- *Makefile.**** – Makefile files for compilation of TRANSIMS components.
- *fstream, iostream, sstream* – special include files for TRANSIMS components.

2. SYSTEM REQUIREMENTS

2.1 Hardware Requirements

TRANSIMS has been tested on Intel hardware with Pentium II processors of 400 MHz or greater. A slower processor can be used but will result in longer execution times for the TRANSIMS programs.

The Output Visualizer uses OpenGL (Mesa). Refer to Appendix A for a list of graphics boards that are supported by OpenGL on Linux.

2.1.1 Memory Requirements

Most of the TRANSIMS components require 128 megabytes or less of memory. Memory requirements for the Output Visualizer depend on the size of the data files. The largest data files are those that contain vehicle snapshot data from the Traffic Microsimulator. Some files as large as 500 megabytes may be generated.

2.1.2 Disk Requirements

Four gigabytes of disk space will support the execution and data collection for TRANSIMS components using the output specifications in the TRANSIMS configuration files supplied with the distribution. Additional disk space may be required if the output collection is increased by the user.

2.1.3 Operating System

TRANSIMS has been tested on Red Hat Linux, Version 5.2. Other Linux implementations could be used, although TRANSIMS has not been tested on them.

Inorder to run the Traffic Microsimulator under PVM or MPI, the Linux kernel must have been compiled with networking support and must have an IP address and host name assigned. An actual network card is not required. The following options must be selected in the Linux kernel configuration:

- Networking support (CONFIG.NET)
- System V IPC (CONFIG_SYSVIPC)
- TCP/IP networking (CONFIG_INET)
- Dummy net driver support (CONFIG_DUMMY) or the appropriate network card driver

The default kernel shipped with Red Hat 5.2 is configured with the appropriate options.

The following package categories should be selected during Red Hat Linux installation in order the run the TRANSIMS components:

- X Window System

- Mesa/GL
- Glut

Additional package categories should be selected in order to compile the TRANSIMS components:

- C Development
- Development Libraries
- C++ Development
- X Development

2.2 Software Requirements

The TRANSIMS distribution requires the following software to be installed:

- X11R6 libraries (*Xmu, Xi, X11, Xext, Xt*)
- OpenGL Utilities Toolkit libraries (*glut*)
- Linux libraries (*stdc++, ld-linux, ICE, SM*)
- Perl

All of the third-party software used by TRANSIMS is available on Red Hat Linux 5.2 distribution CD-ROMs, is supplied as part of the TRANSIMS distribution, or can be downloaded from Red Hat. *Metis, pvm, mpi,* and *xforms* are supplied with the TRANSIMS distribution.

Table 1: Software – Red Hat Packages.*

Name	rpm	location
kernel	2.0.36-0.7	Red Hat 5.2 installation disks
kernel-headers	2.0.36-0.7	Red Hat 5.2 installation disks
kernel	2.0.36-3	ftp://updates.redhat.com/5.2/i386
kernel-headers	2.0.36-3	ftp://updates.redhat.com/5.2/i386
gc	2.7.2.3-14	Red Hat 5.2 installation disks
glibc	2.0.7-29	Red Hat 5.2 installation disks
libg++	2.7.2.8-9	Red Hat 5.2 installation disks
libstdc++	2.8.0-14	Red Hat 5.2 installation disks
make	3.76.1-5	Red Hat 5.2 installation disks
perl	5.004m4-1	Red Hat 5.2 installation disks
XFree86	3.3.2.3-25	Red Hat 5.2 installation disks
Mesa	3.1beta1-3	ftp://contrib.redhat.com/libc6/i386
Mesa-devel	3.1beta1-3	ftp://contrib.redhat.com/libc6/i386
Mesa-glut	3.1beta1-3	ftp://contrib.redhat.com/libc6/i386
Mesa-glut-devel	3.1beta1-3	ftp://contrib.redhat.com/libc6/i386

* rpm -> <Name>-<rpm>.i386.rpm

If you are using another Linux distribution, the third-party software can be obtained from the following URLs.

- Egcs C++ compiler – *http://egcs.cygnus.com/*
- Mesa – *http://www.mesa3d.org/download.html*
- Glut – *http://www-users.cs.unm.edu/~karpis/metis/*
- PVM – *http://www.epm.ornl.gov/pvm/pvm_home.html*
- MPI – *http://www-unix.mcs.anl.gov/mpi/mpich*
- Xforms – *http://bragg.phys.uwm.edu/xforms*
- Perl – *http://language.perl.com/*

3. INSTALLATION

TRANSIMS requires that Linux be installed with the software described in Section 2. The TRANSIMS distribution requires 600 megabytes of disk space.

3.1 Installation Instructions

1) Choose the directory where the distribution will reside. This directory will be referred to as *TRANSIMS_HOME* in the following instructions. Users of TRANSIMS must have read and write permissions in this directory.

2) Mount the CD-ROM. This operation may require *root* permissions. On Red Hat Linux 5.2, the CD-ROM directory is */mnt/cdrom*. This directory may be different on other Linux distributions. The directory where the CD-ROM is mounted will be referred to as *CD_ROM_DIRECTORY* in the following instructions:

```
% /bin/mount <CD_ROM_DIRECTORY>
```

Example:
```
% /bin/mount /mnt/cdrom
```

3) Run the installation script on the CD-ROM using the command:

```
% /bin/sh <CD_ROM_DIRECTORY>/install.sh <CD_ROM_DIRECTORY> <TRANSIMS_HOME>
```

Example:
```
% /bin/sh /mnt/cdrom/install.sh /mnt/cdrom /home/transims
```

This script will copy the TRANSIMS distribution from the CD-ROM to the *TRANSIMS_HOME* directory and install it.

4) Users must have read and write permissions on the *TRANSIMS_HOME* directory and subdirectories. Change the file permissions in the TRANSIMS distribution if necessary.

5) The environment variable *TRANSIMS_HOME* must be set for all users in order to run TRANSIMS.

Bash shell:
Add the following line to your *.profile*
```
export TRANSIMS_HOME=<full path name of directory where TRANSIMS is installed>
```

C Shell:
Add the following line to your *.cshrc*
```
setenv TRANSIMS_HOME <full path name of directory where TRANSIMS is installed.>
```

6) The user must be running X Windows before starting the Output Visualizer program (*Vis*) or the Population Synthesizer program (*Syn*). To start the X server, use the *startx* command (*/usr/X11R6/bin/startx*).

7) TRANSIMS installation is complete. Linux/Intel binaries are distributed on the CD-ROM. No compilation is needed unless you are installing on another platform.

TRANSIMS_HOME/data/samples directory contains tarred and zipped files with sample output from the TRANSIMS components. Refer to Section 4.0 for a description using the TRANSIMS components.

3.2 Compilation

3.2.1 Compilation Requirements

The TRANSIMS distribution contains data and programs for demonstrating the TRANSIMS framework. The distribution also includes the source code for the TRANSIMS modules for examination by the user. Compilation of the source is not necessary to run the components on Linux on an Intel platform. If the source requires recompilation for another operating system or hardware platform, the following software must be installed to compile the TRANSIMS components:

- Linux – kernel headers
- Compiler – Egcs C++ (egcs 1.0.3a or higher)
- Glut Release 3.0 or higher
- Mesa 3.0/GL 3.0
- X11R6 – include files and libraries
- *metis, pvm, mpi*, and *xforms* from TRANSIMS Distribution
- Gnu make

3.2.2 Compilation Instructions

Linux/Intel binaries are supplied with the TRANSIMS distribution. Compilation of the TRANSIMS components is necessary only if you are installing on a different hardware platform/operating system. Use the following instructions to compile the TRANSIMS components, if needed.

1) Verify that the appropriate software described in Section 3.2.1 is installed.

2) If necessary, change the macro definitions in *TRANSIMS_HOME/source/Makefile.SITE* to customize for your site.

3) Change directory to *TRANSIMS_HOME* and run the *buildit* script.

```
% cd $TRANSIMS_HOME
% ./buildit
```

The file *TRANSIMS_HOME/errs* is a log of the build process. Information messages and errors will be reported in this file. If you want to track the progress of the compilation process, edit the *buildit* script and remove the lines that direct output into the logfile, *errs*. TRANSIMS makefiles require Gnu make. Other make utilities will not work with these makefiles. The *buildit* script searches for the *make* utility in */usr/local/bin and /usr/bin* directories.

4. GETTING STARTED

This TRANSIMS distribution provides networks and supporting data to demonstrate the TRANSIMS framework. The major modules of TRANSIMS, including Usage and Tutorial sections, are described in Volume 2—*Software*, Part 1—*Modules*.

Two networks are provided to exercise the TRANSIMS modules and the TRANSIMS framework: the Multimode Network and the Local Streets Network. The Multimode Network supports travel via walking, driving, and riding the bus. Specialized, simplified population and activity generators are given to produce populations and activities for this network. The Local Streets Network allows all of the modules of TRANSIMS to be tested. The purpose of this network is to demonstrate the usage of the full TRANSIMS Population Synthesizer and Activity Generator modules. This distribution also provides two networks that are used to calibrate and test the TRANSIMS microsimulation: the Circle Network and Tee Network.

4.1 Running a Microsimulation Calibration

The Circle Networks are used to calibrate the microsimulation. Each calibration is executed using a preset plan file, and the output is filtered to produce an analysis file containing the statistics of interest. Refer to Volume 2—*Software*, Part 3—*Test Networks* Sections 2 and 3, *Circle* and *Tee Networks*, respectively, for descriptions of the microsimulation calibration networks.

The microsimulation output files from calibration runs are filtered to simplify data analysis. The filters on the freeway, merge, and left-turn calibration runs collect summary data on five-cell sample blocks placed at specified locations on the calibration networks. The data is summarized over three-minute intervals. The sample blocks cut across all lanes of the link, but data is reported on a lane-by-lane basis, as well as the link totals.

Table 2: Calibration Network – sampling locations.

Network	Sample Block Location	Sampled Type
Freeway	Sites 491 – 495 on the circle	Circle vehicles
Merge	Sites 491 – 495 (Link 7) on the circle Sites 501 – 505 (Link 1) on the circle	Circle vehicles Merging vehicles
Left Turn	Sites 491 – 495 (Link 7) on the circle Sites 1 – 5 (Link 11) on exiting link	Circle vehicles Left-turn vehicles

4.1.1 Freeway Calibrations

One- and three-lane freeway traffic is calibrated by moving traffic around the circle on links 1 through 7. Plans for these two calibrations are in files *TRANSIMS_HOME/data/plans/Freeway1.plans* and *TRANSIMS_HOME/data/plans/Freeway3.plans*. These files contain vehicle plans that start vehicles on one of the seven links and move the vehicles around the circle. The density of vehicles is continuously increased by adding more vehicles to the network. The vehicles continue around the circle and are not removed. A small snippet of one vehicle's plans is given below. In this plan, vehicle and individual number 1 start at parking location 4 on link 4, then pass through nodes 5, 6, 7, 1, 2, 3, and 4 in order.

```
1 0 1 1 1 1
2 4 2 1 2
3360358 3360360 1
1 0 1
3361
1 0
5 6 7 1 2 3 4
5 6 7 1 2 3 4
5 6 7 1 2 3 4
5 6 7 1 2 3 4
```

The microsimulation for the two freeway calibrations is carried out using the configuration files, *Freeway1_config* and *Freeway3_config* using the command:

```
perl $TRANSIMS_HOME/scripts/Msim.pl $TRANSIMS_HOME/config/Freeway1_config sim.log
```

or

```
perl $TRANSIMS_HOME/scripts/Msim.pl $TRANSIMS_HOME/config/Freeway3_config s im.log
```

Snapshot output is collected on link 7 and saved in the file *TRANSIMS_HOME/output/Freeway1/freeway1_snap.veh* for the one-lane freeway, and *TRANSIMS_HOME/output/Freeway3/freeway3/snap.veh* for the three-lane freeway calibration. This output is filtered using *TRANSIMS_HOME/bin/FreewayFilter***.** This filter is executed by the following command (where <> denotes input data)

```
$TRANSIMS_HOME/bin/FreewayFilter <# lanes> <snapshot file name> <output file>
```

Example:

```
$TRANSIMS_HOME/bin/FreewayFilter 1 $TRANSIMS_HOME/output/Freeway1/freeway1 snap.veh freeway1.filtered
```

This produces output files with the format in Table 3. The one-lane Freeway calibration takes ~30 minutes to run on a 500 MHz processor. The three-lane Freeway calibration takes ~70 minutes to run.

Table 3: Freeway filter format.

Field	Interpretation
Simulation Time	Seconds since simulation start
Lane	Lane number
Density	Vehicles/km/lane at the sample block on the circle
Flow	Vehicles/hour/lane at the sample block on the circle

The columns contain the vehicle flows and roadway densities on the lanes on link 7. These can be plotted to produce a diagram showing the relationship between the flows and the densities. Sample output from the one-lane Freeway calibration is in the file *TRANSIMS_HOME/data/samples/calibrations/Freeway1.tar.gz*.

4.1.2 Other Calibrations

Refer to Volume 2—*Software*, Part 3—*Test Networks*, Section 2 for instructions on running merge and left-turn calibrations. Refer to Volume 2—*Software*, Part 3—*Test Networks*, Section 3 for instructions on running other microsimulation calibrations.

4.1.2.1 Emissions Estimator on the Tee Network

The TRANSIMS Emissions Estimator can be used to calculate emissions based on the output of the Traffic Microsimulator on the Tee Network. Refer to Volume 2—*Software*, Part 3, *Test Network*, Section 3.3.1 for information on running the Emissions Estimator on sample data from the Tee Network. Sample Tee Network Traffic Microsimulator and Emissions Estimator output is in the file *TRANSIMS_HOME/data/samples/calibrations/Tee.tar.gz*.

4.2 Feedback Iterations Using the Multimode Network

The Multimode Network is used to demonstrate feedback through the TRANSIMS components. The Simplified Population Generator, Simplified Activity Generator, Route Planner, and Traffic Microsimulator are used in the iterations. The first iteration through the components will generate the following if they do not exist:

- synthetic population composed of single-person households located on the Multimode Network
- a TRANSIMS vehicle file for the population
- home-work-home activities for the population
- route plans for the activities
- results of execution of the route plans and the Traffic Microsimulator

Refer to Volume 2—*Software*, Part 3—*Test Network*, Section 4.2 for a description and diagram of the Multimode Network. Refer to Volume 2—*Software*, Part 2—*Selectors* for a description of the feedback process, iteration database, and the use of selectors that will be demonstrated on this network.

The feedback process is initiated and controlled by an iteration script. A sample script is supplied with this distribution in *TRANSIMS_HOME/scripts/expt.sh*. This script will be used to execute TRANSIMS modules on the Multimode Network.

4.2.1 Overview

The purpose of this script is to automate an iteration process. The script

- builds and archives the inputs (populations, vehicles, activities, and plans) for each iteration,
- builds the required indexes,
- runs the Traffic Microsimulator,
- archives some output (currently animation binary files) from each iteration,
- appends to the iteration database summary statistics for each traveler after each iteration, and
- merges new inputs into previous inputs to prepare for the next iteration.

The script can be used with a "clean slate," in which none of the inputs are available, or with previously prepared input files. It maintains separate input archives for different types of

populations (e.g., mass transit, freight, transient, and "household population" or "pop"). Input data for each type can be generated differently. For example, transit schedules can be parsed to generate transit driver plans and vehicles, while the Route Planner is used to generate plans for members of the household population.

4.2.2 Algorithm

For each population type, the script executes the following procedures, which are described in more detail below:

1) Build a population

2) Locate a population

3) Build vehicles

4) Build Activities

5) Build Plans

The results of each of these procedures are placed in a subdirectory of the "current" working directory named *it.<n>*, where *<n>* is the (zero-based) iteration number. This subdirectory is further divided into a subdirectory for each different population type. For example, if the configuration key PLAN_FILE is set to */home/transims/net_plans*, then when the "pop" population type plans are created on iteration 3, they are placed into the file *it.3/pop/net_plans*.

As each result is obtained, it is merged into an index in the "current" subdirectory, which is also subdivided by population type. Thus, in the example above, the new plans would be merged into the index in *current/pop/net_plans*. These indexes contain all of the data from previous iterations that have not been overridden, whereas the ones in *it.<n>* contain only the data generated on the most recent iteration. For efficiency, the plan indexes in the "current" subdirectory are also truncated to the start and end times of the simulation.

Finally, a link is formed between a file in the run directory (current working directory) and the corresponding population-type specific index in the "current" directory. In the example above, *net_plans.trv.idx* would become a link to the file *current/pop/plans.trv.idx*. This is done so that each procedure will have available to it the most recent results of all the previous procedures for that population type.

After every population type has been considered, plan and vehicle indexes are constructed using the indexes in each of the "current" subdirectory's population types. These indexes contain all of the data from every iteration (except that which has been overridden by later iterations) from all the population types. These indexes are the ones used by the Traffic Microsimulator.

After the Traffic Microsimulator runs, the Selector builds the iteration database and selects travelers for activity regeneration and rerouting. On succeeding iterations, the Route Planner and Activity Generator will use the feedback files to update only the required travelers. Also, the Route Planner will use the OUT_SUMMARY_NAME_1 file created in the previous iteration to obtain link travel time delays.

4.2.2.1 Build a Population

If necessary (but not for transit), the *BuildPop* procedure calls the *Popgen* executable after removing the POP_BASELINE_FILE for each population type. It moves the POP_BASELINE_FILE it creates into the appropriate iteration's population type subdirectory. Since *Popgen* is called to build an entire population, not just to modify certain members, the new population file overwrites (via a UNIX link) any previous population in the "current" population type subdirectory, rather than merging into it. A link is also formed between the file in "current" and POP_BASELINE_FILE.

4.2.2.2 Locate a Population

If necessary (but not for transit), the *LocPop* procedure calls the *Poploc* executable after removing the POP_LOCATED_FILE for each population type. It moves the POP_LOCATED_FILE it creates into the appropriate iteration's population type subdirectory. Since *Poploc* is called to locate an entire population, not just to modify certain members, the new population file overwrites (via a UNIX link) any previous population in the "current" population type subdirectory, rather than merging into it. A link is also formed between the file in "current" and POP_LOCATED_FILE.

4.2.2.3 Build Vehicles

If necessary, for the "pop" population type, the *BuildVehicles* procedure calls the *Vehgen* executable. For the "transit" population type, it does nothing, because vehicles will be generated when the transit driver plans are built.

First, the script removes VEHICLE_FILE and its associated indexes, then it generates vehicles. It creates the associated indexes and moves them all to the iteration directory in the appropriate population type subdirectory. Then, since it has generated a complete set of vehicles and not just an update to the previous set, it replaces the vehicle file and indexes in the "current" subdirectory with those in the iteration subdirectory and links VEHICLE file to the ones in the "current" subdirectory.

4.2.2.4 Build Activities

On the first iteration, this procedure runs the LANL Activity Generator, which processes the entire population file and creates activities in the ACT_FULL_OUTPUT file. On subsequent iterations, it runs the LANL regenerator, which processes only those activities listed in ACT_FEEDBACK_FILE and creates activities in the ACT_PARTIAL_OUTPUT file.

The script moves the output and the traveler and household indexes into the appropriate iteration and population type subdirectory, and merges them into the "current" population type subdirectory. Finally, these indexes are linked into indexes for the file specified by ACTIVITY_FILE.

4.2.2.5 Build Plans

Based on the activity file and vehicle file generated in earlier steps, the Route Planner creates a plan for each traveler on the first iteration. On subsequent iterations, the file specified by

ROUTER_HOUSEHOLD_FILE may exist. If so, the Route Planner will generate routes only for the households indicated in that file.

The Route Planner uses free speed delays on links by default. If the file specified by OUT_DIRECTORY and OUT_SUMMARY_NAME_1 exists, the link-specific time delays given in that file are used instead.

After generating the plan file (which will be temporarily placed in PLAN_FILE), the script moves the plan file and a traveler index to the iteration subdirectory, in the appropriate population type subdirectory. The time-sorted index is not created at this point, because it will not be handled properly by *MergeIndices*. The traveler index is merged into the one in the "current" directory for the appropriate population type. The resulting time index is linked to an index for the file specified by PLAN_FILE.

4.2.2.6 Configuration Keys

Configuration Key	Description
BASELINE_FILE LOCATED_FILE VEHICLE_FILE ACTIVITY_FILE PLAN_FILE	These configuration keys specify base names for the indicated files that are used as starting points for iteration 0. For each of the population types "pop" and "transit," the corresponding type's string is added to the base file name. If the resulting file exists, it will be used. Otherwise, the script will call an appropriate tool to generate it. The last component of the path in each of these values will be used as a filename for the corresponding file in the run directory. A new configuration file will be generated in the run directory with the new file name information overriding the old values. For example, suppose PLAN_FILE is */home/transims/plans*. Then, on the 0th iteration, the script will look for */home/transims/plans.transit*. If it exists, it will be used for the transit driver plans. If not, another executable will be called to generate the plans. This executable will not update */home/transims/plans.transit*. Next, the script will look for */home/transims/plans.pop*. If it does not exist, the Route Planner will be called to generate a plan set. The Traffic Microsimulator will look for the file *plans* in the run directory. This file, generated by the iteration script, will contain all of the plans in both *current/pop/plans* and *current/transit/plans*. This allows the re-use of a particular starting point in an experiment, even if it is very expensive to construct all the files that make up the starting point.
EXPT_NUM_ITER	The number of iterations to perform. Default = 1.

4.2.3 Running the Script

The iteration script is given the name of a TRANSIMS configuration file as a command line argument. The configuration file for the Multimode Network is *TRANSIMS_HOME/config/Calnet_default*.

Select a run directory. RUN_DIRECTORY in the following instructions refers to the complete pathname of this directory. The output from the feedback iterations requires ~20 megabytes of disk space in the run directory and ~300 megabytes of space in the *TRANSIMS_HOME/output/Calnet directory*. You must also have read and write permissions in both directories.

Change directory to run directory.

```
% cd <RUN_DIRECTORY>
```

Execute the iteration script from the run directory using the Multimode configuration file, *TRANSIMS_HOME/config/Calnet_default*.

```
% /bin/sh $TRANSIMS_HOME/scripts/expt.sh $TRANSIMS_HOME/config/Calnet_default
```

The iteration script will place information about the iterations in the RUN_DIRECTORY. It will create a subdirectory for each iteration(it.0, it.1, …). Three iterations are specified in the *Calnet_default* configuration file. Three iterations take about one hour to complete on a 500 MHz processor.

Iteration 0 creates a population, home-work-home activities for the population, and route plans for the activities. The execution of the route plans in the Traffic Microsimulator produces information about each traveler's trips. The Selector places pertinent information about the traveler into an iteration database. This information is used to select travelers for activity or route plan changes in future iterations. Subsequent iterations use the selected travelers to regenerate activities, make new route plans, and then execute the plans in the Traffic Microsimulator. Sample output created in the run directory for three iterations is in the file *TRANSIMS_HOME/data/samples/multimode/Calnet.tar.gz*.

4.3 Running TRANSIMS Components on Local Streets Network

The Local Streets Network allows all of the modules of TRANSIMS to be tested. The purpose of this network is to demonstrate the usage of the full TRANSIMS Population Synthesizer and Activity Generator modules and the construction of a more complex network. Additionally, with particular selector scripts, the effects of local streets on traffic behavior may be tested. The network consists of a grid of local streets surrounded by a set of arterial streets and freeways. The local streets contain both a walk layer and a street network. Activity locations are on both the walk layer and the major roads.

Refer to Volume 2—Software, Part 3—*Test Networks,* Section 5.2, for a complete description of the Local Streets Network.

4.3.1 Running the TRANSIMS Modules

The Local Streets Network demonstrates the major modules of TRANSIMS. The programs are located in the *TRANSIMS_HOME/bin* directory. A TRANSIMS configuration file is required by most of the TRANSIMS modules. Sample configuration files for the Local Streets Network are *TRANSIMS_HOME/config/Local_config1, Local_config2*. Sample output from all of the modules is in the file *TRANSIMS_HOME/data/samples/local/Local.tar.gz*.

Select a run directory and invoke the following TRANSIMS modules from this directory. RUN_DIRECTORY in the following instructions refers to the complete pathname of this directory. Approximately 250 megabytes of output will be produced, so choose a run directory on a file system that has enough free space. You must have read and write permissions in the run directory and in the *TRANSIMS_HOME/output/Local* directory.

Change directory to the run directory.

```
% cd <RUN_DIRECTORY>
```

Copy the sample configuration files for the Local Streets Network, *TRANSIMS_HOME/config/Local_config1* and *TRANSIMS_HOME/config/Local_config2*, into the run directory.

```
% cp $TRANSIMS_HOME/config/Local_config1 <RUN DIRECTORY>

% cp $TRANSIMS_HOME/config/Local_config2 <RUN DIRECTORY>
```

Edit the local copy (in your run directory) of *Local_config2*. Change the entries for VEHICLE_FILE and PLAN_FILE by inserting your run directory name between the < > brackets, then remove the < > brackets.

Copy the sample vehicle file for the Local Streets Network (*TRANSIMS_HOME/data/vehicles/local.vehicles*) into your run directory.

```
% cp $TRANSIMS_HOME/data/vehicles/local.vehicles <RUN DIRECTORY>
```

Change the file permissions to read and write on the *local.vehicles* file.

```
% chmod u+rw ./local.vehicles
```

The TRANSIMS modules must be run in the following order:

1) **Population Synthesizer** (*TRANSIMS_HOME/bin/Syn*)
   A baseline synthetic population will be generated from sample census data supplied with this release. PUMA00400 from New Mexico census data is used to generate the population. Refer to Volume 2—*Software*, Part 1—*Modules*, Sections 1.3 and 1.4 for complete instructions (including setting the necessary environment variables) on running the Population Synthesizer. Generation of this small population demonstrates the process by which large populations are created from census data.

   Example:
   ```
   % $TRANSIMS_HOME/bin/Syn
   ```

   The Population Synthesizer creates three population files:

Family Population	<prefix>*Family_Synthetic_HHRecs.out*
Non-Family Population	<prefix>*Non_Family_Synthetic_HHRecs.out*
Group Quarters Population	<prefix>*Group_Synthetic_HHRecs.out*

   <prefix> refers to the string that you typed into the Population Synthesizer text entry window. This string is prepended to the names of the population files.

   These files are combined manually to make a single synthetic population file. Edit the nonfamily (<prefix>*Non_Family_Synthetic_HHRecs.out*) and group quarters (<prefix>*Group_Synthetic_HHRecs.out*) and remove the header lines from each file. The header lines are the first two lines in each file that begin with "Household Demographics:" and "Person Demographics:", respectively.

   Concatenate the three files into a single population file, *mypop.baseline*.

   ```
   % cat <prefix>Family_Synthetic_HHRecs.out > mypop.baseline

   % cat <prefix>Non_Family_Synthetic_HHRecs.out >> mypop.baseline
   ```

```
% cat <prefix>Group_Synthetic_HHRecs.out >> mypop.baseline
```

The combined population file, *mypop.baseline*, should contain the header lines from the family population file and the data lines from all three files. This combined baseline population file will be used to demonstrate how to locate a population on a transportation network and then how to generate a TRANSIMS vehicle file from the located population.

2) **Block Group Locator** (*TRANSIMS_HOME/bin/BlockGroupLoc*)
Each household in the baseline population will be assigned a home location on the Local Streets Network. Each household and person in the population will also be assigned a unique TRANSIMS ID. The Block Group Locator uses the TRANSIMS configuration file, *Local_config1*. The combined population file, *mypop.baseline*, is named in the configuration file. The located population will be placed in the file *mypop.located*.

```
% $TRANSIMS_HOME/bin/BlockGroupLoc Local_config1
```

3) **Vehicle Generator** (*TRANSIMS_HOME/bin/Vehgen*)
The Vehicle Generator creates a TRANSIMS vehicle file that contains an entry for each vehicle in a household. The starting location of the vehicle will be a TRANSIMS parking location that is accessible via the walk network from the household's home location. The Vehicle Generator uses the TRANSIMS configuration file *Local_config1*. The located population, *mypop.located*, is named in this configuration file. The results of the Vehicle Generator will be in the file, *mypop.vehicles*.

```
% $TRANSIMS_HOME/bin/Vehgen Local_config1
```

4) **Activity Generator** (*TRANSIMS_HOME/bin/ActivityGenerator*)
The Activity Generator generates activities for persons in the population. In this release, all modes in activities are car mode. The Activity Generator does not use a TRANSIMS configuration file; instead a special configuration file is used—*TRANSIMS_HOME/config/atp.config*. The activities that are generated will be in the file *local.act*. Refer to Volume 2—*Software*, Part 1—*Modules*, Section 2 for a description of the Activity Generator.

```
% $TRANSIMS_HOME/bin/ActivityGenerator $TRANSIMS_HOME/config/atp.config
```

5) **Route Planner** (*TRANSIMS_HOME/bin/Router*)
The Route Planner creates route plans on the Local Streets network from the activities generated by the Activity Generator. Refer to Volume 2—Software, Part 1—*Modules*, Section 3 for a description of the Route Planner. The Route Planner uses the TRANSIMS configuration file *Local_config2*. The vehicle file, *local.vehicles*, will be used by the Route Planner to coordinate with the activities produced by the Activity Generator.

```
% $TRANSIMS_HOME/bin/Router Local_config2
```

6) **Traffic Microsimulator** (*TRANSIMS_HOME/bin/PVM.ARCH.LINUX/CA*)
The Traffic Microsimulator simulates the movement and interactions of travelers in the transportation system. Using a plan provided by the Route Planner, each traveler attempts to execute that plan on the transportation system. The combined traveler interactions produce emergent behaviors such as traffic congestion. Refer to Volume 2—*Software*, Part 1—*Modules*, Section 4 for a description of the Traffic Microsimulator. The microsimulation is

invoked from a Perl script that initializes the communication mechanism (*TRANSIMS_HOME/scripts/Msim.pl*). You must know the location of the Perl executable on your system. The Perl script (*Msim.pl*) requires the full path name of a TRANSIMS configuration file as a command line argument. The Traffic Microsimulator uses the TRANSIMS configuration file, *Local_config2*. The vehicle file, *local.vehicles*, will be used by the Traffic Microsimulator to coordinate with the plans produced by the Route Planner (*local.plans*).

The Traffic Microsimulator program resides in a subdirectory of *TRANSIMS_HOME/bin*. Since the executable depends on the type of parallel communication that will be used, Parallel Virtual Machine (PVM), or Message Passing Interface (MPI), two executables are provided in the following subdirectories:

TRANSIMS_HOME/bin/ARCH.PVM.LINUX
and
TRANSIMS_HOME/bin/ARCH.MPI.LINUX.ch_p4

The user may specify the parallel communication mechanism in the TRANSIMS configuration file. PVM is the default and is used in the configuration files supplied with this release.

```
% /usr/bin/perl $TRANSIMS_HOME/scripts/Msim.pl <RUN_DIRECTORY>/Local_config2 sim.log
```

The microsimulation results will be in the *TRANSIMS_HOME/output/Local* directory, which is specified in the configuration file using the key, OUT_DIRECTORY. The simulation log file is *sim.log* in your run directory. The Traffic Microsimulator takes ~35 minutes to run on a 500 MHz processor.

7) **Output Visualizer** (*TRANSIMS_HOME/bin/Vis*)
Refer to Volume 2—*Software*, Part 1—*Modules*, Section 6 for a description and instructions on using the Output Visualizer to display the output of the Traffic Microsimulator and Route Planner. The Output Visualizer uses the same configuration file that was used by the Traffic Microsimulator, *RUN_DIRECTORY/Local_config2*. The output of the Traffic Microsimulator is in the *TRANSIMS_HOME/output/Local* directory. The name of the vehicle snapshot file is *local_snap.veh*.

```
% TRANSIMS_HOME/bin/vehtobin $TRANSIMS_HOME/output/local/local_snap.veh local_snap.veh.bin

% TRANSIMS_HOME/bin/Vis Local_config2
```

Appendix A

Hardware and Software Requirements of the Output Visualizer

Graphics Boards for TRANSIMS Visualizer

At the time of this writing, the XFree86 server supported the following graphics boards: All of the following graphics boards have Linux drivers available for them and are hardware accelerated at least to some extent.

Table 4: Supported graphics boards.

Manufacturer & Model	Price	RAM	Speed	$/Performance
3D Labs Oxygen GMX	$2300	96MB	2.56	898.44
ATI 3D Rage Pro	$ 130	8MB	0.74	175.68
Diamond FireGL 1000 Pro	$ 130	8MB	1.02	127.45
Diamond Viper 330	$ 79	8MB	1.00	79.00
Diamond Viper 550	$ 127	16MB	1.65	76.97
Diamond Stealth II	$ 64	8MB	1.13	56.64
ELSA Gloria Synergy	$ 98	8MB	1.17	83.76
ELSA Gloria-L	$1117	16MB	1.98	564.14
ELSA Gloria-XL	$1073	16MB	1.98	541.92
ELSA Gloria-XXL	$1328	16MB	1.98	670.71
ELSA Erazor II	$ 135	16MB	1.62	83.33
Hercules Thriller 3D	$ 120	8MB	1.01	118.81
Matrox Millenium	$ 202	4MB	1.02	198.04
Matrox Mystique G200	$ 99	8MB	1.00	99.00
Number 9 Revolution 3D	$ 96	8MB	1.38	69.57
STB Velocity 128	$ 65	8MB	1.13	57.52

NOTE: The Speed column is calculated through various sets of benchmarks. There is no one source of benchmarks for all boards; therefore, the numbers can only be approximate and should not be taken as absolutes. Also, benchmarks tend to be from Windows NT workstations so results may vary. The speed estimates are intended for comparison purposes only.

Also, note that the XFree86 drivers have widely varying degrees as to how well each accelerated driver is written; consequently, a much slower board may actually perform faster than an expensive board due to a more developed server. Also, support does not mean that all modes are implemented into the server—for example, a board that includes 16MB of RAM may only be capable of working in 256 color mode.

Hardware accelerated X servers are also available for 3dfx Voodoo chipset based graphics boards. The quality of these X servers is probably the best available at the time of this writing.

Finally, PC mail order companies are beginning to ship workstations with Linux pre-installed. Among them are:

- Dell – Offers a Pentium III/Diamond Permedia II 8MB AGP based system for approximately $5,000.

- Intergraph Computer Systems – Offers pre-installed Linux computers (Their new Intense 3-D Wildcat 4000 graphics board is probably the fastest 3-D graphics board tested to date for Intel based systems; no X server is available for it at the time of this writing).

In addition, there are at least 90 or more vendors, which can be found by the following URL: *http://www.linux.org*.

Output Visualizer

Software packages required for the Output Visualizer

- Mesa3D – An OpenGL work-alike library.

- GLUT (OpenGL Utilities Toolkit release 3.6)

Both packages are available as rpms for Red Hat Linux, or they can be downloaded from the following URLs:

- GLUT can be downloaded from the following web URL: *http://reality.sgi.com/mjk_asd/glute3/glute3.html*. Click on the *Obtaining the GLUT source code distribution* link. Click on the *(GLUT 3.6)* link to download GLUT release 3.6.

- Mesa3D can be downloaded from the following web URL: *http://www.mesa3d.org/download.html*.

Graphics Board X Server Requirements

There are several sources for OpenGL compatible X servers for 3-D graphics boards running under Linux. These sources include:

- XFree86 – The list of X servers available can be found *at* *http://ftp.xfree86.org/cardlist.html*,and there is no cost associated with obtaining these servers.

- Metrolink Inc. – Produces and sells X-Servers, in particular, Metro Extreme3D (a hardware accelerated OpenGL X-Server for high-end graphics boards is under development at this time. The URL for Metro Link products is: *http://www.metrolink.com/productindex.html*.

- Xi Graphics – Produces and sells X-Servers and is currently developing a 3-D Technology Demo package that will also provide 3-D hardware accelerated display servers for high-end graphics boards.

- *http://glide.xxedgexx.com* – An author who contributes to Linux Journal updates these pages regularly. This site provides useful pointers in obtaining X-Servers.

- *http://www.linuxberg.com* – A freeware/shareware listing of applications/tools/utilities/servers available for download for the Linux operating system.

The absolute minimum resolution for the Output Visualizer is 1024 x 768; however, 1280 x 1024 is recommended to minimize frustration in viewing multiple windows simultaneously. Also, select the highest number of bits per pixel when configuring the X Server. Eight bits-per-pixel is enough; the quality of 3-D lighted displays improves with more bits per pixel.

Appendix B:

Running on Solaris/SPARC

The TRANSIMS-LANL Version 1.0 distribution must be recompiled in order to run on Solaris. The following instructions assume that the environment variable *TRANSIMS_HOME* has been set to the top level directory where TRANSIMS-LANL is installed. You must have read and write permissions in the TRANSIMS_HOME directory to compile the TRANSIMS components.

Third-party Software

Compiler

TRANSIMS is compiled with the egcs C++ compiler, egcs 1.1.2 or higher, which can be downloaded from the URL: http://egcs.cygnus.com/

Gnu Make

TRANSIMS components must be compiled using the Gnu make utility. Make is available from ftp://ftp.gnu.org/pub/gnu/.

XForms

The TRANSIMS Synthetic Population Generator uses xforms, which is available from *http://bragg.phys.uwm.edu/xforms*. Download the Solaris/Sparc version, bxform-088.tgz. Unzip and untar the distribution in a temporary directory (TMP_DIR).

Change directory to the xforms/FORMS subdirectory.

```
% cd <TMP_DIR>/xforms/FORMS
```

Remove the old copy of *forms.h* from TRANSIMS_HOME/source/SYNPOP/include/SUN

```
% rm -f $TRANSIMS_HOME/source/SYNPOP/include/SUN/forms.h
```

Copy *forms.h* to TRANSIMS_HOME/source/SYNPOP/include/SUN

```
% cp ./forms.h $TRANSIMS_HOME/source/SYNPOP/include/SUN
```

Make the directory for the xforms library.

```
% mkdir $TRANSIMS_HOME/source/SYNPOP/lib
```

```
% mkdir $TRANSIMS_HOME/source/SYNPOP/lib/SUN
```

Copy the xforms library to the new directory.

```
% cp ./libforms.a $TRANSIMS_HOME/source/SYNPOP/lib/SUN
```

OpenGL and GLUT

The Output Visualizer requires OpenGL and OpenGL Utilities Toolkit (GLUT). OpenGL 1.1.1 for Solaris is available from Sun Microsystems at the following URL: *http://www.sun.com/software/graphics/OpenGL/*. System patches for Solaris that must be installed for OpenGL and lists of the supported graphics boards/framebuffers are also available on this Web site.

GLUT can be downloaded from the following URL:

http://reality.sgi.com/mjk_asd/glut3/glut3.html

Click on "Obtaining the GLUT source code distribution" link. Click on the GLUT 3.6 link to download GLUT, release 3.6. Instructions for installation of each GLUT package are included in the downloaded files.

Recompiling for Solaris/Sparc

The third party software in Section 1.1 must be installed.

1) Remove the TRANSIMS_HOME/bin directory.
   This directory contains executables compiled for a Linux/Intel platform. The directory will be recreated during the compilation.

   ```
   % rm -rf $TRANSIMS_HOME/bin
   ```

2) Compile metis-4.0 for the Solaris/Sparc platform.
   Change directory to TRANSIMS_HOME/metis-4.0 and read the instructions in the file, *INSTALL*. Follow the instructions to compile metis-4.0 for the Solaris/Sparc platform.

3) Compile PVM for the Solaris/Sparc platform.
   Change directory to TRANSIMS_HOME/pvm/pvm3. Set the environment variable, *PVM_ROOT*, to *TRANSIMS_HOME/pvm/pvm3*. *PVM_ROOT* must be set before building or running PVM. Use the PVM architecture SUN4SOL2 even on a Sun multiprocessor machine instead of the shared memory architecture SUNMP.

   Instructions on how to build PVM are in the file *Readme* in the BUILDING AND INSTALLING section. A C compiler and make utility must be in your path in order to build PVM.

4) Compile MPI for the Solaris/Sparc platform.
   Change directory to TRANSIMS_HOME/mpich. The README file has instructions on compilation of MPI. Run the MPI configure script in this directory using the arguments *-arch=solaris -device=ch_p4*. A C compiler and make utility must be in your path in order to build MPI.

   ```
   % cd $TRANSIMS_HOME/mpich
   % ./configure -arch=solaris -device=ch_p4
   % make
   ```

   The TRANSIMS components will use PVM by default so use of MPI is not necessary to run

TRANSIMS. If you plan to specify MPI when running the TRANSIMS components, setup the util/machines/machines.solaris file with the machine names that you will use.

5) Customize macro definitions in TRANSIMS_HOME/source/Makefile.SITE.
Change directory to TRANSIMS_HOME/source and change the file permissions to make Makefile.SITE writable.

```
% cd $TRANSIMS_HOME/source
% chmod u+w Makefile.SITE
```

Edit the file, *Makefile.SITE* to customize for your location.

Comment out the definition of SUNMATH_HOME on line 15.

Original:
```
SUNMATH_HOME := /sw/CSUNWspro/SC4.2/lib
```

Change to:
```
#SUNMATH_HOME := /sw/CSUNWspro/SC4.2/lib
```

Change the following macro definitions to define the location of the egcs C++ (g++) and C (gcc) compilers and the archive command on your system.

G++_EXEC -- Full path name of C++ compiler (g++) on your system, e.g., /usr/local/bin/g++

GCC_EXEC -- Full path name of C compiler (gcc) on your system, e.g., /usr/local/bin/gcc

AR_EXEC -- Full path name of the archive command (ar) on your system, e.g., /usr/ccs/bin/ar.

The following macro definitions are required to compile the Output Visualizer.

OPENGL_HOME -- The directory where the OpenGL libraries and include files are installed, e.g. /usr/openwin.

GLUTHOME - The directory where the GLUT libraries and include files are installed, e.g., /usr/local/glut

Note: If you do not have hardware that supports OpenGL, you can remove the TRANSIMS Visualizer from the compilation. Edit the file TRANSIMS_HOME/source/Makefile.main. Comment out line 417 in this file that adds the Visualization subsystem to the compilation list. A '#' character at the beginning of the line indicates a comment.

TRANSIMS_HOME/source/Makefile.main

Original:
```
DIRS += SYNPOP TBX THREADS TRANSIT VEH ENV
DIRS += VIS
```

Change to:

```
DIRS += SYNPOP TBX THREADS TRANSIT VEH ENV
#DIRS += VIS
```

6) Edit TRANSIMS_HOME/buildit script.
TRANSIMS components must be compiled using the Gnu make utility. If gnu make on your system is not /usr/local/bin/make or /usr/bin/make, the buildit script must be edited to define the location of gnu make.

Change directory to TRANSIMS_HOME and change the file permissions to make buildit writable.

```
% cd $TRANSIMS_HOME
% chmod u+w ./buildit
```

Edit the buildit file to define the location of gnu make on your system.

Remove lines 7 and 8 of this file:

```
echo "Can't find make"
exit
```

and replace with the definition of gnu make on your system:

```
set MAKE=<full path name of gnu make>
```

```
#!/bin/csh
if (-x /usr/local/bin/make) then
    set MAKE=/usr/local/bin/make
else if (-x /usr/bin/make) then
    set MAKE=/usr/bin/make
else
===> Remove these lines and replace with definition of location of
gnu make
    echo "Can't find make"
    exit
===>
endif
endif
```

7) Edit TRANSIMS source files to prevent compilation errors.

TRANSIMS_HOME/source/ROUTER/style.h
TRANSIMS_HOME/source/VIS/convcars.C

Change the file permissions so that the files are writable.

```
% chmod u+w $TRANSIMS_HOME/source/ROUTER/style.h
% chmod u+w $TRANSIMS_HOME/source/VIS/convcars.C
```

TRANSIMS_HOME/source/ROUTER/style.h

```
% cd $TRANSIMS_HOME/source/ROUTER
```

Edit *style.h* and change line 54.

Original:

```
#ifdef SUN  // MR 12-08-98
#  include "/opt/SUNWspro/include/sunmath.h"   /*  for infinity()
*/
#  define INFTY infinity()
#else
```

```
#  include <values.h>
#  define INFTY (MAXFLOAT+MAXFLOAT)
#endif
```

Change to:

```
#ifdef __SUNPRO_CC
#include "/opt/SUNWspro/include/sunmath.h"   /*  for infinity()
*/
#define INFTY infinity()
#else
#include <values.h>
#define INFTY (MAXFLOAT+MAXFLOAT)
#endif
```

TRANSIMS_HOME/source/VIS/convcars.C

```
%cd $TRANSIMS_HOME/source/VIS
```

Edit convcars.C, lines 218, and 297, change ifndef statement to an ifdef statement.

Original line 218:

```
#ifndef LINUX
       theta = atan2pi( y2 - y1, x2 - x1 ) ;
       thetadf =  atan2d( y2 - y1, x2 - x1 ) ;
```

Change to:

```
#ifdef __SUNPRO_CC
        theta = atan2pi( y2 - y1, x2 - x1 ) ;
        thetadf =  atan2d( y2 - y1, x2 - x1 ) ;
```

Original line 297:

```
#ifndef LINUX
           xcar = xlane + distance * cospi( theta ) ;
           ycar = ylane + distance * sinpi( theta ) ;
```

Change to:

```
#ifdef __SUNPRO_CC
           xcar = xlane + distance * cospi( theta ) ;
           ycar = ylane + distance * sinpi( theta ) ;
```

8) Compile the TRANSIMS components.
Change directory to TRANSIMS_HOME. Execute the buildit script to compile the TRANSIMS components. You must have read and write permissions in the TRANSIMS_HOME directory.

```
% cd $TRANSIMS_HOME
% ./buildit
```

Messages from the compilation are directed to the file TRANSIMS_HOME/errs. The compilation will create the directories TRANSIMS_HOME/bin, TRANSIMS_HOME/obj, and TRANSIMS_HOME/lib.

9) To run TRANSIMS components, you must include /usr/openwin/lib in the path defined by the environment variable, LD_LIBRARY_PATH.

10) See Volume 6, Section 4, "Getting Started" for instructions on running the TRANSIMS components. Perl is required to run the TRANSIMS components and can be downloaded from http://language.perl.com/